STAT 542: Final Project

Shuyu Jia (shuyuj2), Yevgeniy Onegov (ykalini2), Zhehui Chen (zhehuic2)

Due May 15, 2020

Contents

1 Introduction

The goal of our project is to find the most vulnerable population to COVID-19 virus and to infer on the possible ways to reduce mortality rate. We use strategies discussed in class to model the time-dependent outcomes and predict them at a future time point.

Project is split into two parts: supervised and unsupervised learning. Unsupervised Learning (UL) methods are leveraged in exploratory analysis to automatically identify structure in the data without using explicitly provided labels. UL methods are also used to represent data with less variables i.e. dimension reduction. We address potential missing data problems before performing UL. The three UL methods we apply are K-means clustering, Hierarchical clustering and a combination of t-SNE and spectral clustering methods for solving three specific questions. Supervised Learning (SL) methods are leveraged to approximate the mapping function which predicts future cases and deaths caused by COVID-19. The main considerations are model complexity and the Bias-Variance tradeoff. The Bias-Variance tradeoff is a balance between Bias, which is the constant error term, and Variance, which is the amount by which the error may vary between different training sets. We implement four different classification models, including Random Forest, Decision Tree, K-nearest neighbors and Boosting, where we map input to output labels while properly tuning parameters. We also accurately implement four different regression models where we map input to a continuous output while properly tuning parameters. Our regression models are Linear Regression, Random Forest, Support Vector Machine and Stochastic Gradient Boosting.

For unsupervised learning section, we show that counties are clustered based on demographic and health features as well as COVID-19 growth pattern. Each cluster has characteristics that are different from other groups. In classification section, we perform a binary classification of the severity of each county with decent accuracy. In regression section, the results show that linear regression model makes the most accurate predictions of the future death counts, even though the root mean squared error (RMSE) gets larger as the days move on. Based on our analysis, we are able to find the most vulnerable counties to the virus as well as give suggestions on reducing the number of death through statistical analysis.

2 Literature review

The research goal of the paper[1] published by Prof. Bin Yu's group is "to both provide access to a large data repository (that combines data collected by a range of different sources) and to provide a predictor to forecast short-term COVID-19 mortality at the county-level in the United States." Combined data from January 22,2020, to April 8,2020 is stored on Github[2] for easy access. At the county level, the data includes COVID-19 cases/deaths from USA Facts and NYT along with demographic information, health resource availability, COVID-19 health risk factors and social mobility information. The paper focuses on predicting the number of confirmed deaths instead of confirmed cases, since confirmed cases fail to capture spread of the virus due to the limitation of testing.

This team performs five models. The first one models each county separately capturing the log-linear relationship between time and number of deaths; the second one models the log-linear relationship between the death count of two consecutive days; the third one is the expansion of the second model with more predictors including the number of cases k days ago, the number of deaths of the neighbor county several days ago and the number of cases of the neighbor county several days ago. The fourth model is also similar to the second one, but with more demographic and healthcare-related features such as median age, population density, number of ICU beds, etc. The fifth one models each county separately capturing the linear relationship between time and the death count. This team finally uses a combination (CLEP) of the third model and the fifth model to determine their results. They find out the majority of counties appear to share an exponential growth, but some are growing sub-exponentially, and that is why a proportion of the linear model come into play.

Lastly, the paper compares their results to the related work and analyzes the reason why it can be challenging

[1]https://www.stat.berkeley.edu/~binyu/ps/papers2020/covid19_paper.pdf
[2]https://github.com/Yu-Group/covid19-severity-prediction

to create good intervals that can account for all the changes. On one hand, the nature of virus itself is highly dynamic; on the other hand, human behavior and policy changes can also be major contributors to influence the trend of both the death and case counts. However, we can still hope for the best that our data and our model can capture the short-term tendency of the COVID-19 deaths at the county level to help people make better decisions at this critical time in this pandemic.

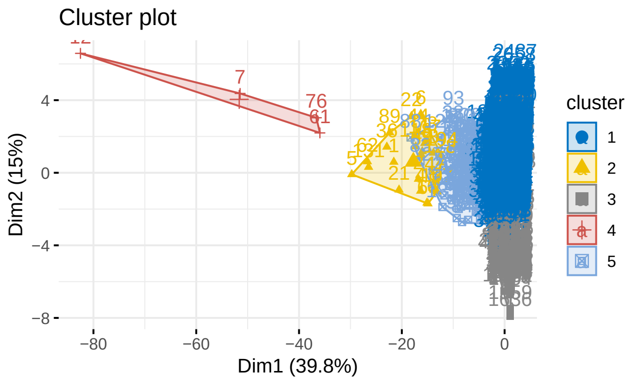
3 Unsupervised learning

At the county level, our data contains 3141 observations (counties) with 276 features including COVID-19 cases/deaths from USA Facts and NYT, automatically updated every day, along with demographic information, health resource availability, COVID-19 health risk factors, and social mobility information.

Before clustering analysis, we first deal with missing values. We define a function "count_na" which is used to calculate the number of missing values for each feature. Variables with over 50% missing data are removed. For other features with acceptable missing rate, we fill the missing based on the mean value of its state since the same state is more likely to have same pattern than using the country mean. After filling, there is no missing values in our data set.

3.1 Q1: Underlying clusters of counties based on the demographics and health-related information (Method: K-means)

To obtain underlying clusters (at county level) in terms of demographics and health-related features, we perform k-means clustering method. K-means method minimizes within-cluster variances (squared Euclidean distances) and it's easy to implement. We first determine the number of clusters (K) by plotting the relationship between total within sum of square and cluster number. We choose 5 as the optimal K since it has smaller "wss" with relatively small value.



The cluster result is shown above. The same color represents underlying clusters of counties with similar demographics and health-related information. Also, We look at the underlying features of each cluster then get the following result:

Cluster 1 has low PopulationEstimate2018, low PopulationEstimate65.2017, low PopulationDensityperSqMile2010, highest DiabetesPercentage, highest HeartDiseaseMortality, highest StrokeMortality,highest Smokers_Percentage, highest RespMortalityRate2014.

Cluster 2 has high PopulationEstimate2018, high PopulationEstimate65.2017, highest PopulationDensityperSqMile2010, high X.EligibleforMedicare2018.

Cluster 3 has low PopulationEstimate2018, low PopulationEstimate65.2017, low PopulationDensityperSqMile2010, low X.EligibleforMedicare2018, high Smokers_Percentage, lowest X.Hospitals, lowest X.ICU_beds.

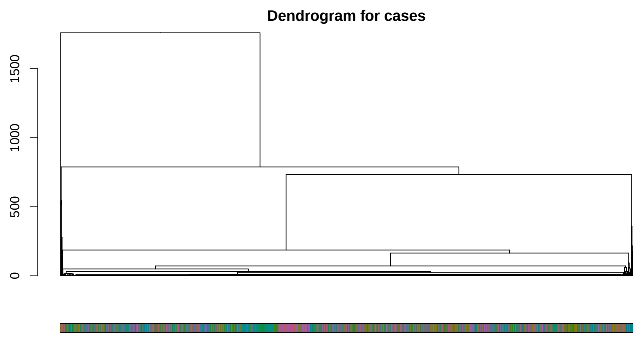
Cluster 4 has highest PopulationEstimate2018, highest PopulationEstimate65.2017, highest X.EligibleforMedicare2018, highest X.Hospitals, highest X.ICU_beds.

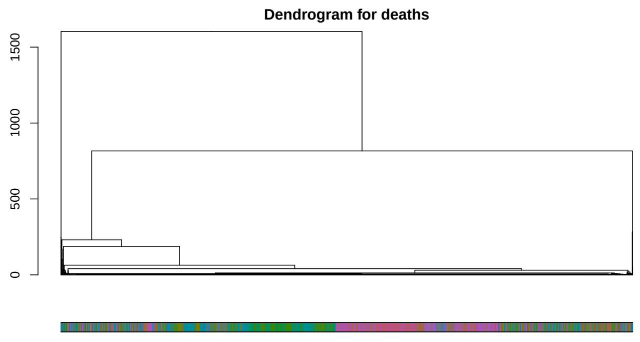
Cluster 5 has average values.

3.2 Q2: Underlying pattern (at the county level) in terms of how the COVID-19 counts are growing (Method: Hierarchical)

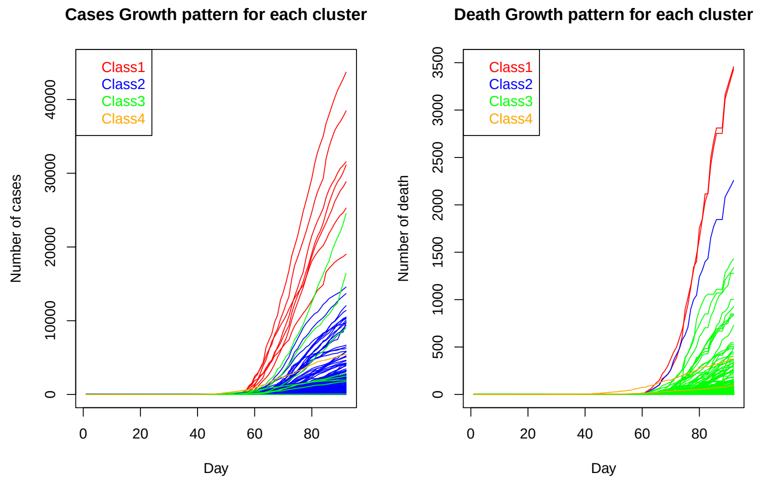
The clustering part of how COVID-19 is growing is based on the number of cases and deaths from January 22,2020 to April 22,2020. Same clusters contain counties that have similar growth pattern. In other words, counties with small distance in the number of growth will be formed into one group. In the following analysis, we analyze case and death pattern separately. We implement hierarchical clustering method for this part. Especially, for the measurement of distance, we choose the "dtw" method representing "Dynamic Time Warping", which is an algorithm calculates the least cumulative distance alignment between points of two time-series data.

In order to select the best number of clusters (K) for both case and death problem, we generate the plot about the relationship between K and total within group sum of square. The best K should not be large, but results in small within group distance. According to the figures we generated, we choose 4 as the number of clusters of both case and death clustering. The dendrogram for death result shows how cluster tree grows and the same color in the color bar below the dendrogram represents the same state.





The dendrogram for cases (top) and deaths (bottom) results shows how cluster tree grows and the same color in the color bar below the dendrogram represents the same state. The height of each split represents how separated the two county clusters are. The figure shows that the first two clusters have much more cases and deaths than counties in other clusters.



The plot shows the growth curve for both cases and deaths of all 3141 county and the same color represent the same cluster.

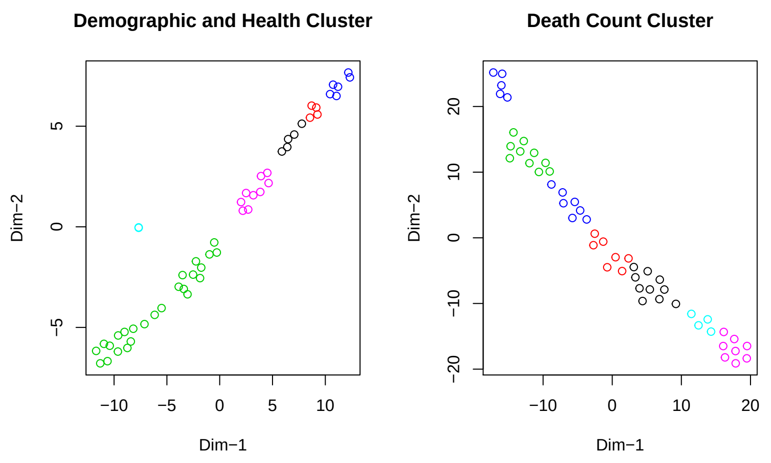
The figure on the left shows the pattern of cases for each cluster and we could see that different clusters have different maximums and break time. Combined with the cluster size table, we could see most counties are grouped into class2, which represents most cities have similar growth pattern without significant characteristics. The class1 contains counties such as New York, Kings, Queens, Nassau, etc., which all locate in NY, are the regions most affected by the epidemic. Class3 contains counties such as Cook(IL), Los Angeles(CA), San Diego(CA), etc., which are big cites in each state. King(WA) is the only county in class4, which is the origin of the outbreak.

The second figure on the right shows the pattern of deaths for each cluster and we could see that different clusters have different maximums and break time. Combined with the cluster size table, we could see most counties are grouped into class3, which represents most cities have similar growth pattern without significant characteristics. Class1 contains counties including Kings(NY) and Queens(NY), with the most deaths and the most significant increasing rate. Class2 contains counties Bronx(NY), with less deaths than the first cluster, but with similar growth pattern. King(WA) and Santa Clara(CA) are the counties in class4, which are the first two of death.

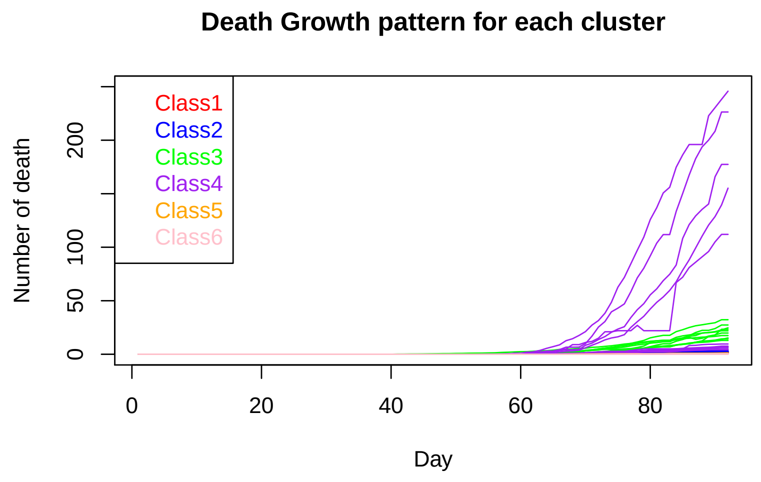
3.3 Q3: Underlying pattern (at the state level) in terms of the combination of demographic and health as well as how COVID-19 death count grows. (Method: t-SNE + Spectral)

t-SNE[3] can give really nice results when we want to visualize many groups of multi-dimensional points. Once the 2D graph is done we might want to identify which points cluster in the t-SNE blobs. The COVID-19 data set is high dimensional so t-SNE could significantly help to reduce dimension into two features. Then, we perform spectral clustering method to find the similar states based on the dimension reduction result.

[3]https://jmonlong.github.io/Hippocamplus/2018/02/13/tsne-and-clustering/



The two figures plot the dimension-reduced coordinates of the fiftyone states and the same color represents states in the same cluster. The figure on the left is the result of demograhphc and health clustering and the right one is the result of death count clustering.



The figure above shows the growth curve of all the states in different clusters.

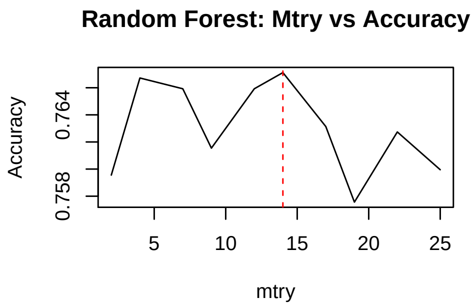
4 Supervised learning

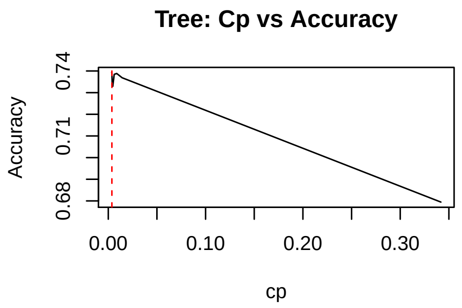
4.1 Classification Models

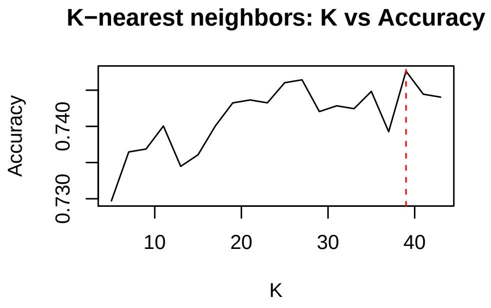
The purpose of this section is to perform a binary classification of "if_severe" variable we are creating (value 1 represents death per 100,000 population is greater than 1 and otherwise 0) using the demographics and health-related information. All of our models are trained using `caret` package[4].

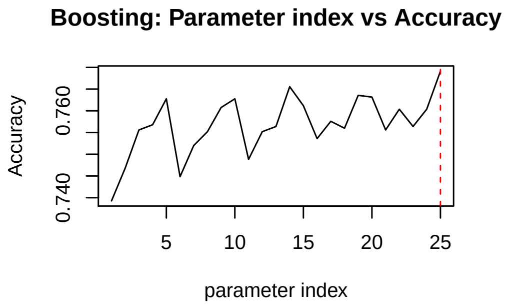
Before doing classification, we need to deal with missing data. We use non-parametric imputation method for the rest of the missing data by implementing "rfImpute" method. This method fills in missing values, then runs Random Forest. Then for missing continuous values, Random Forest computes the proximity-weighted average of the missing values. Then this process is repeated several times.

[4]https://topepo.github.io/caret/model-training-and-tuning.html









For classification problem, we build four models containing Random Forest, Decision Tree, K-nearest neighbors and Boosting. We use function “train” from package “caret” to train and tune parameters in this part.

Random Forest is an ensemble learning method which make prediction by constructing a multitude of decision trees at training time and outputting the class that has the most vote. For the tuning parameter, we tune mtry (number of variables randomly sampled as candidates at each split). Figure (1,1) shows that the RF model has the highest accuracy when mtry=14.

Decision tree breaks down a dataset into smaller and smaller subsets based on decision rules while at the same time an associated decision tree is incrementally developed. Leaves represent class labels and branches represent conjunctions of features that lead to those class labels. For the tuning parameter, we tune cp (complexity parameter that controls the size of the decision tree). Figure (1,2) shows that the Tree model has the highest accuracy when cp=0.004.

The k-nearest neighbors algorithm (KNN) is a non-parametric method for classification that an object is classified by a plurality vote of its nearest k neighbors. For the tuning parameter, we tune k (number of neighbors). Figure (2,1) shows that the KNN model has the highest accuracy when k=39.

AdaBoost method is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers where more weights are adjusted to incorrectly classified instances.For the tuning parameter, we tune iter(number of iteration), and maxdepth (the depth of the base tree). Figure (2,2) shows that the AdaBoost model has the highest accuracy when iter=250 and maxdepth=5.

Model	Accuracy
Random Forest	0.759936
Tree	0.755167
KNN	0.745628
Boosting	0.759936

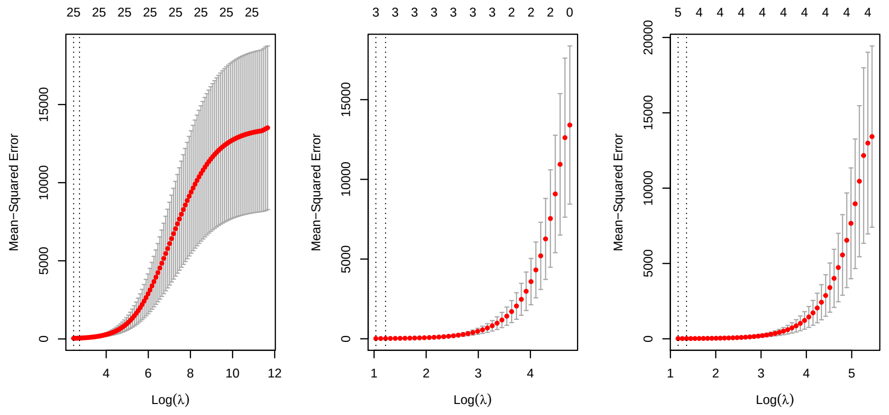
The accuracy table shows that both RF and Boosting model have best classification accuracy among the four models with accuracy 0.7599364.

4.2 Regression Models

The purpose in this section is to perform a regression to predict the number of death one week from April 22,2020. Available predictors not only include demographic and health-related data but also all the death counts and case counts from January 22,2020 to April 22,2020. The predictors we are using include the death counts of previous four days of the target date, 21 demographic and health-related information such as population estimates, median age, number of ICU beds, etc. Specifically, we choose the death counts in April 22,2020 as the response and the previous four days' death counts as predictors. When we try to predict the death count of a future day (say April 25,2020), the demographic and health-related data remain the same, and then we add the death counts from April 21,2020 to April 24,2020 as predictors. Note that some of them are predictions, and we are using previous predictions to make further predictions. However, given the task that we need to predict seven days after April 22,2020, this is the best we can do. It is like the weather forcasting (using the predicted weather to predict future weather), the inaccuracy of further predictions are acceptable. All of our models are trained using `caret` package[5].

4.2.1 Penalized Linear Model

Similarly to the classification part, we used random forest imputation method to fill in missing data once again. The first model we are considering is penalized linear model. The following three plots show how the mean squared error changes with the penalized coefficient λ. The first one is ridge regression, the second one is LASSO regression, and the last one is elastic net regression (with $\alpha = 0.5$ which is half ridge and half LASSO). We can clearly see that the optimal λ for all three model is zero, which indicates that we do not need any penalty on the coefficients. In other words, the ordinary least square gives us the lowest mean squared error.



4.2.1.1 Ordinary Least Square

The we can perform ordinary least square regression. Since we already figure out the optimal λ in the previous section, we do not need to tune any parameters here. We performed data preprocessing by standarizing all variables before training, and the model was trained using 5-fold cross validation. The result is as follows. The training RMSE is about 1.99.

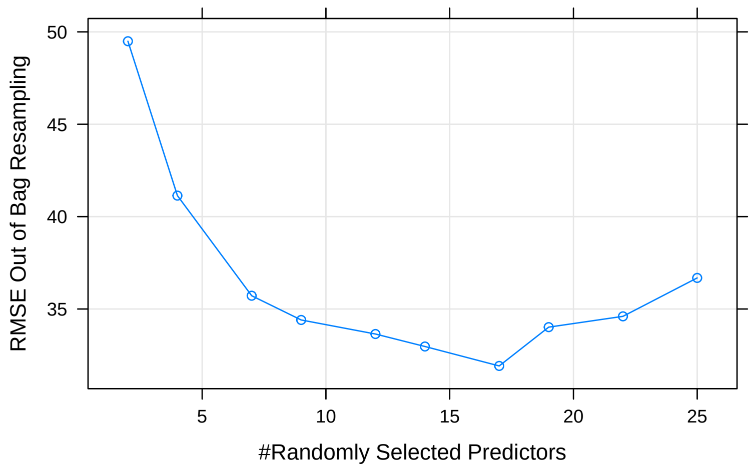
```
## Linear Regression
##
## 3141 samples
##   26 predictor
##
## Pre-processing: centered (25), scaled (25)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2513, 2513, 2512, 2513, 2513
## Resampling results:
```

[5]https://topepo.github.io/caret/model-training-and-tuning.html

```
##
##   RMSE      Rsquared   MAE
##   1.993677  0.9994779  0.4444501
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

4.2.2 Random forest

```
## Random Forest
##
## 3141 samples
##   25 predictor
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared
##    2    49.49321  0.8188272
##    4    41.13591  0.8748462
##    7    35.71858  0.9056395
##    9    34.40674  0.9124434
##   12    33.64628  0.9162710
##   14    32.97031  0.9196015
##   17    31.91786  0.9246524
##   19    34.01815  0.9144100
##   22    34.60280  0.9114427
##   25    36.68289  0.9004757
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 17.
```

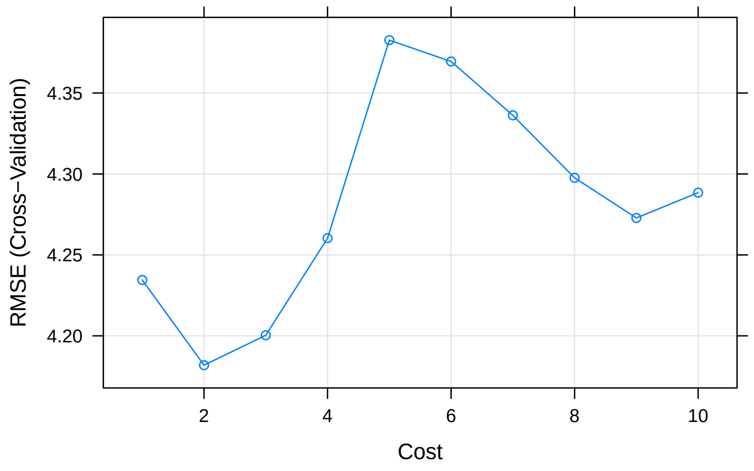


The second model is random forest. Here we used root mean squared error as our metric, and the best “mtry” parameter was chosen among 10 different values by default grid. The “mtry” parameter correspondes to how many variables are we considering in each split in each tree model. This model is trained using out-of-bag validation, the best tuning paramter is 17. The plot above shows how RMSE changes for different “mtry” values.

4.2.3 Support Vector Machines with Linear kernel

```
## Support Vector Machines with Linear Kernel
```

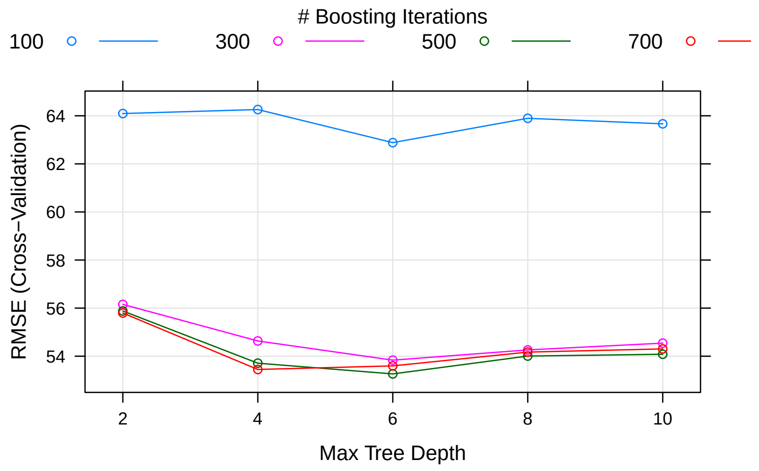
```
##
## 3141 samples
##   25 predictor
##
## Pre-processing: scaled (25)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2513, 2513, 2512, 2513, 2513
## Resampling results across tuning parameters:
##
##   C   RMSE      Rsquared   MAE
##    1  4.234502  0.9968080  3.168054
##    2  4.181876  0.9968016  3.094630
##    3  4.200357  0.9967797  3.102354
##    4  4.260348  0.9966723  3.199891
##    5  4.382675  0.9965810  3.329345
##    6  4.369485  0.9966072  3.317507
##    7  4.336232  0.9966475  3.283695
##    8  4.297612  0.9966911  3.240181
##    9  4.272820  0.9966800  3.204164
##   10  4.288461  0.9966431  3.216656
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was C = 2.
```



The third model is support vector machines with linear kernel. In this model, the tuning parameter C is penalize coefficient for the sum of ξ, which is the distance between the support vector and its corresponsing margin (we have covered that in lecture). We scale all the variables before training since SVM is sensitive to distancing. The best parameter C is chosen among 10 different values from 1 to 10, and the model is validated using 5-fold cross validation. As a result, the best tuning paramter is 2.

4.2.4 Stochastic Gradient Boosting

```
## Stochastic Gradient Boosting
##
## 3141 samples
##   25 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2513, 2513, 2512, 2513, 2513
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##    2                 100      64.09318  0.7948456  10.096797
##    2                 300      56.15585  0.7964647   7.935246
##    2                 500      55.88104  0.8122784   7.981852
##    2                 700      55.79584  0.8258631   8.010197
##    4                 100      64.25964  0.7942247   9.083706
##    4                 300      54.63470  0.8137564   6.611231
##    4                 500      53.71037  0.8366945   6.617226
##    4                 700      53.44862  0.8528354   6.545363
##    6                 100      62.88345  0.8032307   8.800238
##    6                 300      53.83571  0.8200422   5.933377
##    6                 500      53.26388  0.8411343   5.998329
##    6                 700      53.59749  0.8522399   6.013995
##    8                 100      63.89554  0.8001324   8.834473
##    8                 300      54.26185  0.8188218   5.832159
##    8                 500      54.00670  0.8399818   5.877124
##    8                 700      54.17098  0.8523283   6.023134
##   10                 100      63.66563  0.7931361   8.816222
##   10                 300      54.54337  0.8166290   5.863925
##   10                 500      54.08131  0.8347375   5.805085
##   10                 700      54.30323  0.8507234   5.942680
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.01
## Tuning
##  parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 500, interaction.depth = 6,
##  shrinkage = 0.01 and n.minobsinnode = 10.
```



The fourth model is stochastic gradient boosting. Here we consider four values (100,300,500,700) of n.trees, which represents the number of gradient boosting iteration. Increasing n.tree reduces the error on the training set, but setting it too high may lead to over-fitting. The interaction.depth parameter is the maximum number of nodes for each tree starting from a single node, and we consider five values (2,4,6,8,10). The shrinkage parameter is just the learning rate, and we set it constant here at a level of 0.01. Since this is a decently large dataset, we keep it at a relatively small number. The n.minobsinnode parameter is the minimum number of observations in trees' terminal nodes. (If you are familiar with Python packages, this is usually what they called max_leaf.) Here we set it at a constant level of 10.[6] As before, we validate this model using 5-fold cross validation to select the best combination of tuning parameters. The best combination of tuning parameters are interaction.depth = 6 and n.trees = 500, while holding shinkage=0.01 and n.minobsinnode = 10. The above plots show how the RMSE changes while using different combinations of interaction.depth and n.trees.

4.2.5 Test our models using real data (Bonus)

Having all four models, we used them to predict the future seven days from April 23,2020 to April 29,2020, and compared the predictions with the updated data. The resulted accuracy (RMSE) table is as follows.

Model	best tuning	04/23	04/24	04/25	04/26	04/27	04/28	04/29
OLS	lambda = 0	8.7696	10.0988	11.8913	12.7437	13.9549	16.5113	19.2373
RF	mtry = 17	10.4739	11.7490	14.0104	15.5999	17.4183	20.4339	23.6675
SVM	C = 2	9.1918	11.2006	13.4895	14.9007	16.4730	19.1129	21.8546
SGB	depth = 6, ntree = 500	16.8469	23.2811	25.7510	26.9234	27.9113	28.8432	30.2579

We can see that the best model is linear model, second place is SVM with linear kernel, third place is random forest, the worst one is gradient boosting. Also the RMSE gets larger and larger for further predictions, which is as expected. Take Champaign County in IL as an example (since we all stuck in here for now), the real death count and predictions are as follows:

```
##            death_423 death_424 death_425 death_426 death_427 death_428 death_429
## true_value  4.000000  4.000000  5.000000  5.000000  5.000000  5.000000  6.000000
## lm_pred     4.337966  4.610611  4.830561  5.008000  5.151145  5.266623  5.359782
## svm_pred    3.140504  2.535660  2.110019  1.810487  1.599700  1.451365  1.346979
## rf_pred     4.002400  4.002400  4.002400  4.002400  4.002400  4.002400  4.002400
## gbm_pred    3.476649  2.494424  1.626080  1.626080  1.626080  1.626080  1.626080
```

4.2.6 Discussion, Improvements and Reflection

Method-wise, up to this point we only considered to predict the death count using the data from previous four days plus some extra features. However, we do not know whether using four days' data is appropriate. It could be more than we need, or it could not be enough. There are several improvements we could consider. The first one is that we can make is that we can treat this as a tuning parameter and try to see how the result changes when we are using different length of period as predictors. The second one is to try some transformations of linear model such as log-linear, or even consider a combination of them just like Prof.Yu's team suggested. The third one is that we could predict the increment of covid-19 from the previous day rather than predicting the count directly since there is some situations in our result that the value of the next day is smaller than the previous day.

Data-wise, the dataset we are using was updated in April 23,2020. If you go to see the most recent updates, you will find out that some of the death counts in our data set are underestimated. To reflect on this, one of the reasons why linear model performs the best in our prediction is due to the lack of update to our dataset. According to the literature, some of the counties have shown a sub-exponential growth already, and the lack of updates just makes the linear model even better. However, this might not be the case in the reality. Thus, we could have a better shot at predicting the future if we have a more accurate dataset, but this is the best we can do when we are in April 22,2020.

Admittedly, even if we fix all the issues mentioned above, it may still not be enough to capture all the changes, because the world is constantly changing, in both human behaviors and policy changes, for the same reasons as we discussed in the literature review section. Given that, we should still try our best to capture the short-term trends and to help people make better decisions at this critical time in this pandemic.

[6] https://www.listendata.com/2015/07/gbm-boosted-models-tuning-parameters.html

5 Collaborator's Questions

5.1 Q1: What population is the most vulnerable to this virus?

We define the most vulnerable population as a population with the highest increase of COVID-19 cases and mortality rate over the given period of time.

From a county perspective, we have already done a clustering analysis on both cases and death data. Kings, Queens, Bronx, Nassau, New York, Suffock and Westchester fall into the cases cluster with the highest increase of cases. Kings and Queens fall into a death cluster with the highest increase of deaths. Thus, the most vulnerable population is residing in Kings and Queens counties. To be more general, cities in NY are more likely to be vulnerable population.

```
##                                 Most Vulnerable Less Vulnerable
## PopulationEstimate2018               2097956.00    102249.60389
## PopulationEstimate65.2017             297612.00     15922.50988
## PopulationDensityperSqMile2010         29608.70       231.14458
## CensusPopulation2010                 2040176.67     96432.50478
## X.EligibleforMedicare2018             325998.67     19541.51177
## X.FTEHospitalTotal2017                 29079.67      1820.99522
## X.ICU_beds                               239.00        23.44551
```

From a feature perspective, the table above compares some features with significant difference between the most vulnerable population and less vulnerable ones. Result shows that vulnerable counties have much more population density, easier access to healthcare providers (including hospitals) and than less vulnerable ones.

5.2 Q2: What can we do to reduce mortality?

Recall our clustering result of death count, we first look at the underlying features of the COVID-19 data cluster that contains Queens and Kings counties. They both fall into cluster 2.

This cluster has the highest population density per square mile, second highest population, heart disease mortality, PopulationEstimate65.2017, X.FTEHospitalTotal2017, X.EligibleforMedicare2018, MedicareEnrollment.AgedTot2017, X.Hospitals, X.ICU_beds and average values for other variables. We look at the statistically significant variables from the analysis before and compare them to the underlying features of the cluster. All of the underlying features of this cluster are highly significant in the linear model. Cluster 4 has higher values for most of the variables but has a much lower population density.

Then we run the best regression model from the analysis before on this particular cluster. We look at the statistically significant variables and compare them to the underlying features of the cluster, the top six coefficients (except for intercept and day count) ordered by absolute value are shown below:

```
##                         Feature       Coefficient
## 1          CensusPopulation2010  5.54690086890599
## 2 MedicareEnrollment.AgedTot2017  5.10426342958921
## 3        PopulationEstimate2018 -4.48048893159666
## 4     X.EligibleforMedicare2018  -2.9725502807191
## 5     PopulationEstimate65.2017  -1.9558270306933
## 6                    X.ICU_beds -1.47302525237037
```

Regression coefficients are estimates of the unknown population parameters and describe the relationship between a predictor variable and the response. In our case, the response is the number of death and predictor variables are demographic and health related features. The positive sigh of coefficient suggests the response increases as the variable increase and vise versa. According to the coefficient table, we could see that to predict number of death, the population is positively related to death count, while the increase of EligibleforMedicare and number of ICU beds will reduce the growing pattern.

5.2.1 Suggested Actions

Lowering the density population is one way to reduce deaths. This is done through "stay at home" order to reduce the spread of COVID-19. A lot of significant variables are connected to hospitals. Higher percentage of testing due to higher amounts of hospitals means higher COVID-19 numbers. We can stop testing done to reduce COVID-19 cases. On the other hand, higher amounts of hospitals and ICD beds means more people getting treated from the virus. Thus, death numbers will go down.